Disciplina: Projeto e Otimização de Algoritmos

Professor: Rafael Scopel

Exercício Greedy

- Este trabalho consiste no desenho, análise e implementação de algoritmos greedy e de divisão e conquista.
- O trabalho pode ser realizado grupos de até 4 alunos.
- O trabalho deve ser implementado em Java.

<u>Objetivos do trabalho:</u>

<u>Problema 1</u>

Você e um grupo de amigos deseja realizar um rally pelo deserto de Dakkar. Os regulamentos da corrida determinam que os times podem viajar apenas durante o dia. Após analisar o mapa seus colegas identificaram um grande conjunto de bons pontos de para passar a noite. Durante várias sessões de brainstorming eles (sim, você estava estudando para as provas P1 e não participou) definiram o seguinte algoritmo para determinar se vocês conseguem chegar no próximo ponto de parada antes do anoitecer.

Algoritmo: Cada vez que vocês chegarem a um potencial ponto de parada, vocês determinam se conseguem chegar no próximo ponto de parada antes do anoitecer. Se vocês conseguirem, vocês continuam dirigindo, caso não consigam, vocês param e acampam no ponto atual.

Eles afirmam que o algoritmo acima os levará a linha de chegada com o menor número de paradas. Você concorda?

Para tornar a questão mais precisa vamos assumir as seguintes premissas:

- Modelaremos a trilha do rally como um longo segmento de linha de comprimento L.
- Vocês conseguem viajar no máximo d quilômetros por dia antes de anoitecer.
- Assumiremos que os pontos de parada estão localizados a distâncias $x_1, x_2, \dots, x_n$ do ponto de partida.
- Assumiremos também que os seus amigos sempre estão corretos quando estimam se conseguem ou não chegar ao próximo ponto de parada antes do anoitecer.
- Vamos considerar um conjunto de pontos de parada como válidos se a distância entre cada par adjacente é no máximo d, e o primeiro ponto de parada está a no máximo uma distância d do início e o último ponto de parada está a uma distância no máximo d do final da corrida. Portanto, um conjunto de pontos de parada é valido se vocês conseguirem acampar nestes pontos e ainda completar toda a trilha.
- Assumimos que o conjunto n com todos os pontos de parada é valido.

Com base nas informações acima, podemos afirmar que o algoritmo proposto é ótimo, no sentido de que encontra o menor conjunto de pontos de parada válidos que completa o rally?

Estruture a resposta no formato de um relatório com as seguintes sessões:

1. O Problema;
2. O Algoritmo;
3. Análise do Algoritmo;

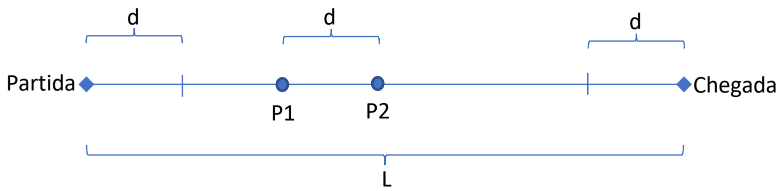
4. Implementação e Tempo de Execução;

Sugestão de Resposta:

Normalmente um algoritmo greedy parece correto quando o observamos pela primeira vez, porém antes de sucumbir ao apelo intuitivo do algoritmo, precisamos perguntar:

- Por qual motivo o algoritmo pode não funcionar?
- Com quais aspectos deveríamos nos preocupar?

Existe uma preocupação natural com esse algoritmo: Não ajudaria pararmos mais cedo em algum dia específico, de maneira a sincronizar possíveis paradas futuras? Mas se você refletir sobre isso, irá se perguntar se isso poderia realmente ocorrer. Poderia existir uma solução que intencionalmente fica atrás da solução greedy, e posteriormente acelera de maneira a ultrapassar a solução greedy? Como seria possível ultrapassar, dado que a solução greedy viaja o máximo possível a cada dia?



A última consideração é muito semelhante ao contorno do argumento baseado no princípio de "manter-se a frente" que vimos durante as aulas. Talvez, seja possível demonstrar que contanto que a estratégia greedy esteja a frente num dado dia, nenhuma solução pode alcançar e ultrapassar a solução greedy no próximo dia.

Agora precisamos transformar a consideração acima numa prova demonstrando que o algoritmo é ótimo, identificando um sentido natural em que os pontos de parada escolhidos pelo algoritmo "estão a frente" de qualquer outro conjunto de pontos de parada. A diferença do exemplo visto em sala da aula é que para o problema do Escalonamento de Tarefas queríamos maximizar uma determinada quantidade e no problema atual queremos minimizar uma quantidade, ou seja, o número de paradas.

Deixe $R = \{x_{p_1}, \dots, x_{p_k}\}$ denotar o conjunto de pontos de parada escolhidos pelo algoritmo greedy, e suponha que por meio de contradição que existe um conjunto menor de pontos de parada válidos. Chamaremos esse conjunto menor de $S = \{x_{q_1}, \dots, x_{q_m}\}$, com $m < k$.

Para obter uma contradição, primeiramente demonstramos que o ponto de parada alcançado pelo algoritmo greedy em cada dia j é mais distante que o ponto de parada atingido por qualquer solução alternativa, ou seja:

A.1 Para cada $j = 1,2, \dots, m$, temos que $x_{p_j} \geq x_{q_j}$.

Prova:

Provamos por meio de indução em j. O caso $j = 1$ segue diretamente da definição do algoritmo greedy: seus amigos viajam o mais longe possível no primeiro dia antes de parar para dormir. Agora assuma o caso $j > 1$ e assuma que a afirmação A.1 é verdadeira para todos $i < j$. Então:

$$x_{q_j} - x_{q_{j-1}} \leq d$$

como S é um conjunto válido de pontos de parada, e

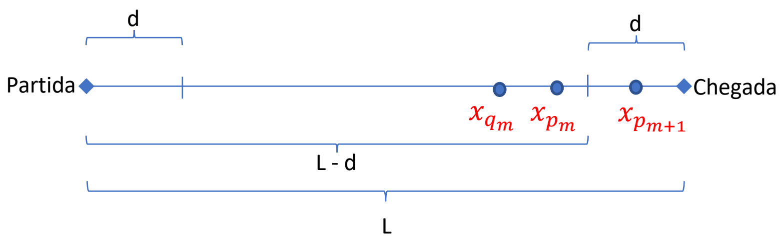
$$x_{q_j} - x_{p_{j-1}} \leq x_{q_j} - x_{q_{j-1}}$$

como $x_{p_{j-1}} \geq x_{q_{j-1}}$ pela hipótese de indução. Combinando essas duas inequações, temos:

$$x_{q_j} - x_{p_{j-1}} \leq d$$

Isso significa que os seus amigos têm a opção de viajar de $x_{p_{j-1}}$ para x_{q_j} em um dia; e então a localização x_{p_j} na qual eles irão finalmente acampar só pode ser mais distante que x_{q_j} ou ser o próprio x_{q_j}. Similar a prova do Escalonamento de Tarefas, aqui também o algoritmo greedy está se mantendo a frente, pois, a cada passo, a escolha feita pela solução alternativa é um dos seus pontos válidos.

A afirmação A.1 implica em particular que $x_{q_m} \leq x_{p_m}$. Agora, se $m < k$, então devemos ter $x_{p_m} < L - d$, pois de outra forma seus amigos nunca precisariam ter parado na localização $x_{p_{m+1}}$.

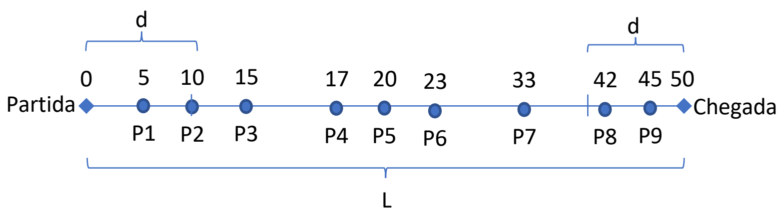


Combinando essas duas inequalidades,

$$x_{q_m} \leq x_{p_m} < L - d$$

concluímos que $x_{q_m} < L - d$; mas isso contradiz a afirmação que S é um conjunto válido de pontos de parada.

Consequentemente, não podemos ter $m < k$, e então provamos que o algoritmo greedy produz um conjunto válido de pontos de parada com o menor tamanho possível.



O exemplo acima define uma instância do problema com $n = 9$ pontos de parada, a distância máxima que pode ser percorrida é $d = 10$ e o tamanho total do rally é $L = 50$. O algoritmo greedy seleciona os seguintes pontos como solução ótima. No primeiro passo seleciona P2, P5, P6, P7 e P8. O conjunto $R = \{x_{p_2}, x_{p_4}, x_{p_6}, x_{p_7}, x_{p_8}\}$ consiste de soluções válidas, pois o primeiro e último ponto estão a uma distância d do inicio e final da corrida e os pontos internos estão a no mínimo uma distância d entre si.

Algoritmo:

DakkarCrossing$(n, x_{p_1}, x_{p_2}, \ldots, x_{p_n}, d, L)$
$R \leftarrow \emptyset$
$lastStop = 0$
$i = 0$
While $(lastStop + d < L \;\; AND \; i < n)$
    If $(x_{p_i} - lastStop > d)$
        $lastStop = x_{p_{i-1}}$
        $R \leftarrow R \cup \{x_{p_{i-1}}\}$
    $i = i + 1$

If $(lastStop + d < L)$
    Return $\emptyset$

Return R

Se assumirmos que os pontos do trajeto já estão ordenados por distância, o algoritmo de DakkarCrossing é $O(n)$.